**Lab3实验报告**

**实验内容分析：**

在本次lab中，实现了对于上个lab的扩展与补全，实现了初级版本的扫雷游戏，具体补充了用户交互功能，本次lab中，除了main函数外我实现了initBoolMap，isSweepOut，SweepOneMine这三个函数，并且重载了printMap函数，沿用了上一个lab中的MakeMap函数。

**实现思路：**

在上个lab已经写好了makeMap函数的基础上，我又定义了一个boolean类型的二维数组outerMap，用于存储每个对应位置是否被用户扫过的信息。initBoolMap用于将该数组全部初始化为true表示初始没有任何一个方块被用户翻开。进入循环中，首先打印出提示信息提示用户输入要翻开的行列坐标，然后调用SweepOneMine函数对用户指定位置的信息进行相应更改，同时SweepOneMine的返回值Boolean表明该次扫雷是否点到地雷，如果返回true，则表明踩中地雷，返回提示语的同时打印出innerMap（地图的具体形式），如果返回false，表明没有踩中地雷，继续循环。循环的结束条件是有isSweepOut函数返回，表明是否清空非地雷区域，清空后则打印“恭喜，扫雷完成”的提示信息，程序结束。

**各函数具体实现：**

**makeMap函数：**上个lab实现，直接沿用。功能为随机初始化地雷，并初始化其他位置。

**initBoolMap函数：**将定义的Boolean[][]outerMap全部初始化为true，表示为初始用户没有进行扫雷，每一个位置的情况为未知。

**printMap函数：**这个函数有两个重载，作用为打印地雷地图，其中一个为现阶段用户可见的地图，另一个为实际的地雷地图。

**isSweepOut函数：**功能为判断是否已经全部扫完地雷。通过遍历所有非地雷点看对应的outerMap是否为false，如果存在一个非地雷点没有别扫出，则返回false表示没有完全扫出所有地雷。

**SweepOneMine函数：**用于翻开一个用户指定的位置，返回的Boolean值表示此次是否碰到地雷。

**实验难点：**

采用了另外定义一个Boolean类型的二维数组outerMap来表示每一个位置是否被翻开的状态信息，便于根据用户的输入实时调整。

**附录（具体代码实现）：**

public static void main(String[] args) {

int row,column,num;

Scanner scn=new Scanner(System.in);

System.out.println("请依次输入行、列以及地雷数");

row=scn.nextInt();

column=scn.nextInt();

num=scn.nextInt();

char [][]innerMap=new char[row][column];

boolean [][]outerMap=new boolean[row][column];//true---mask false----unmask

initBoolMap(outerMap);

makeMap(innerMap,num);

printMap(innerMap,outerMap);

int flag=1;

while(!isSweepOut(innerMap,outerMap))

{

System.out.printf("请输入1-%d 1-%d来进行扫雷\n",row,column);

int inputRow=scn.nextInt();

int inputColumn=scn.nextInt();

if(SweepOneMine(innerMap,outerMap,inputRow-1,inputColumn-1))

{

System.out.println("踩中地雷，游戏结束");

printMap(innerMap);

flag=0;

break;

}

else {

printMap(innerMap,outerMap);

}

}

if(flag==1)

System.out.println("恭喜，扫雷完成");

}

public static void makeMap(char [][]a,int num)

{

Random generator=new Random();

for(int i=0;i<num;)

{

int m1=generator.nextInt(a.length),m2= generator.nextInt(a[0].length);

if(a[m1][m2]!='\*')

{

a[m1][m2]='\*';

i++;

}

}

for(int i=0;i<a.length;i++)

{

for(int j=0;j<a[0].length;j++)

{

int val=0;

if(a[i][j]!='\*')

{

for(int k=Math.max(i-1,0);k<=Math.min(i+1,a.length-1);k++){

for(int t=Math.max(j-1,0);t<=Math.min(j+1,a[0].length-1);t++){

if(a[k][t]=='\*'){

val++;

}

}

}

a[i][j]=(char)(val+'0');

}

}

}

}

public static void initBoolMap(boolean [][]a)

{

for (boolean[] booleans : a) {

Arrays.fill(booleans, true);//表示全部被覆盖中

}

}

public static void printMap(char[][]a,boolean [][]b)

{

for(int i=0;i<a.length;i++)

{

for(int j=0;j<a[0].length;j++)

{

if(b[i][j])

{

System.out.print("# ");

}

else

{

System.out.print(a[i][j]+" ");

}

}

System.out.println();

}

}

public static void printMap(char[][]a)

{

for(char[]arr:a)

{

for(char i:arr)

{

System.out.print(i+" ");

}

System.out.println();

}

}

public static boolean isSweepOut(char [][]innerMap,boolean[][]outerMap)

{

for(int i=0;i<innerMap.length;i++)

{

for(int j=0;j<innerMap[0].length;j++)

{

if(innerMap[i][j]!='\*'&&outerMap[i][j])

return false;

}

}

return true;

}

public static boolean SweepOneMine(char [][]innerMap,boolean[][]outerMap,int row,int column)

{

//未防止重复点击、越界等的输入

outerMap[row][column]=false;

if(innerMap[row][column]=='\*')

{

return true;

}

else return false;

}//true----bomb,false----continue